Figure 1 (Prior art)

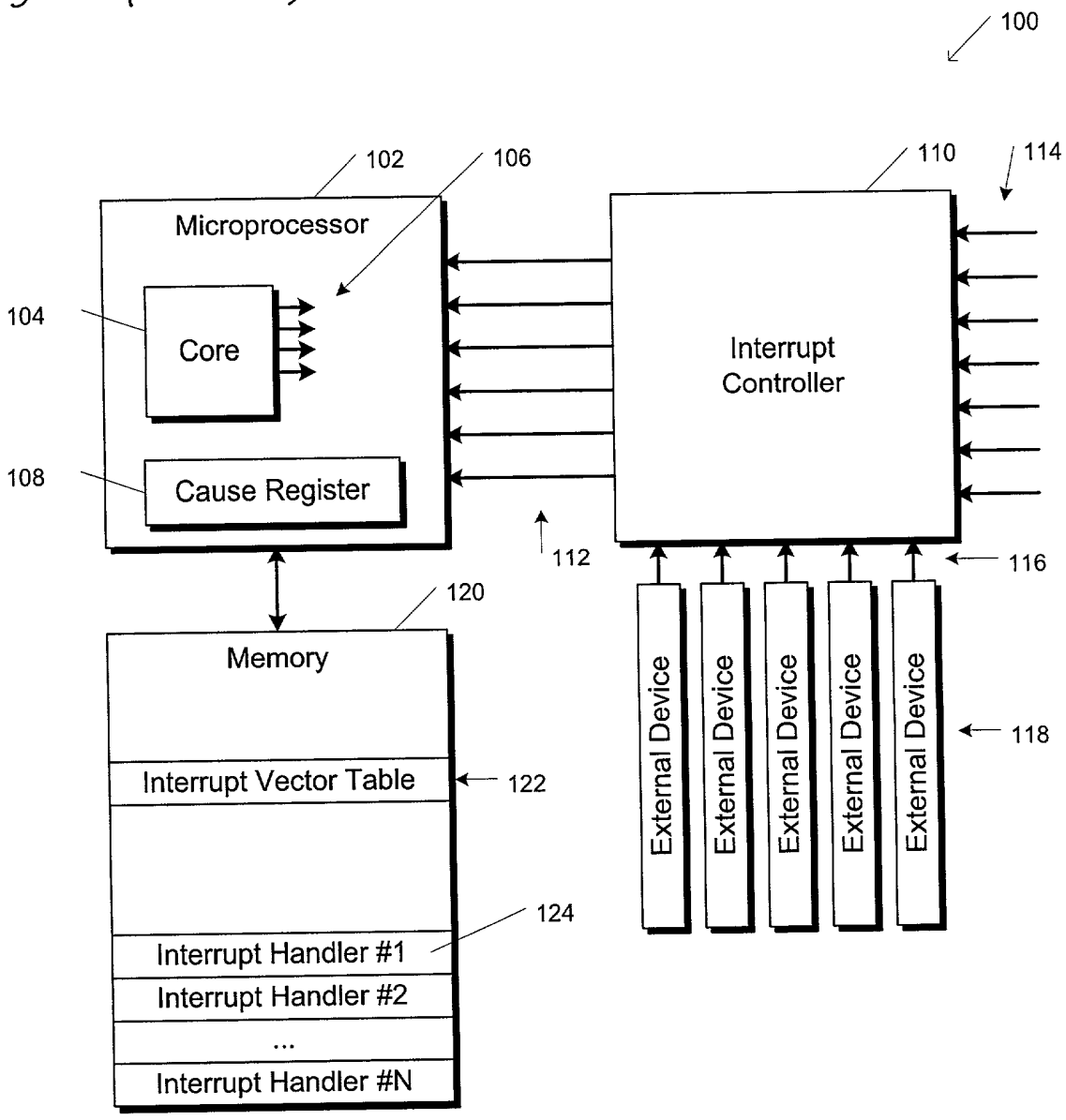


Figure 2 (Prior Art)

Flow Chart for Prior Art Interrupt Handling

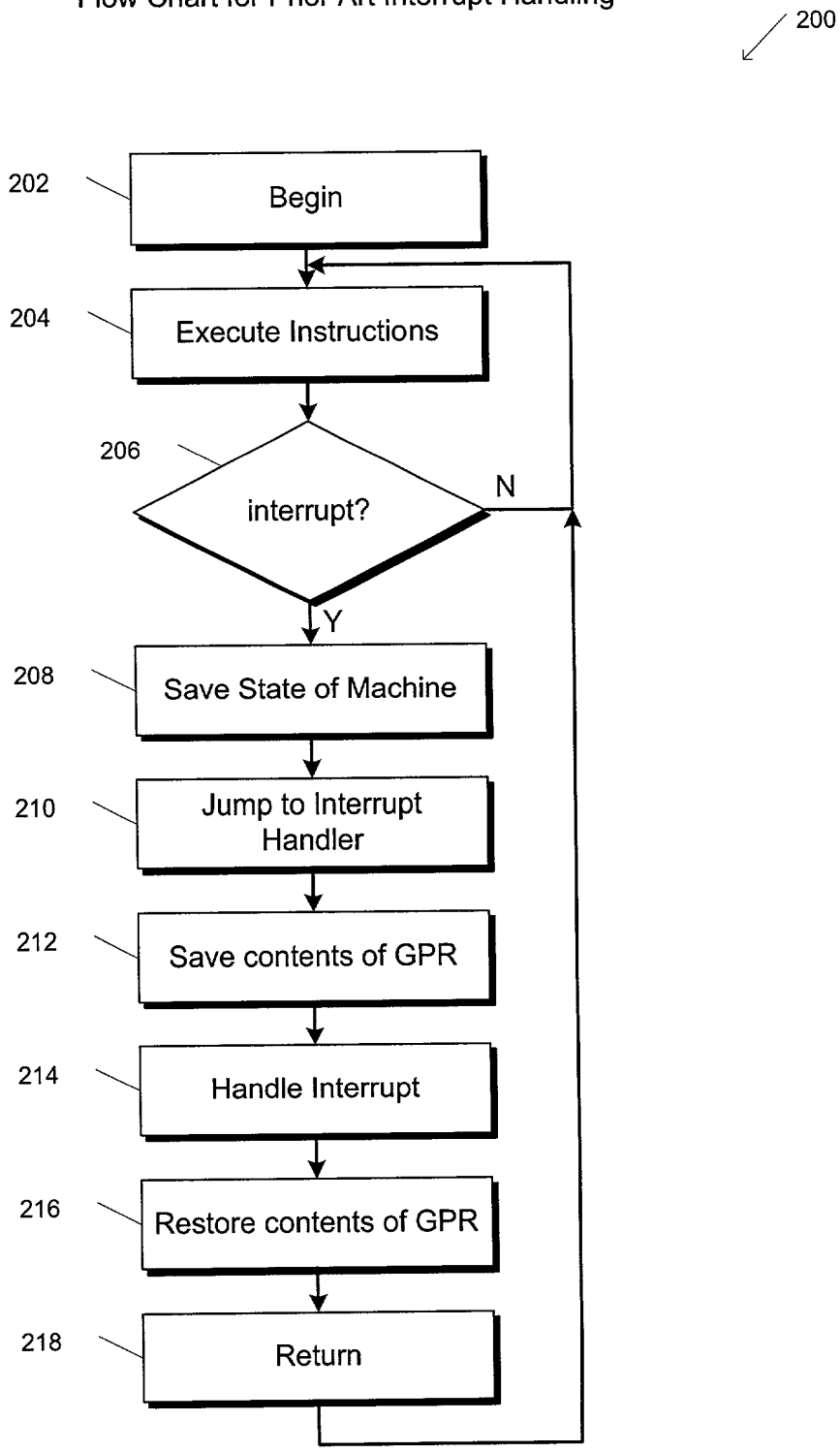


Figure 3

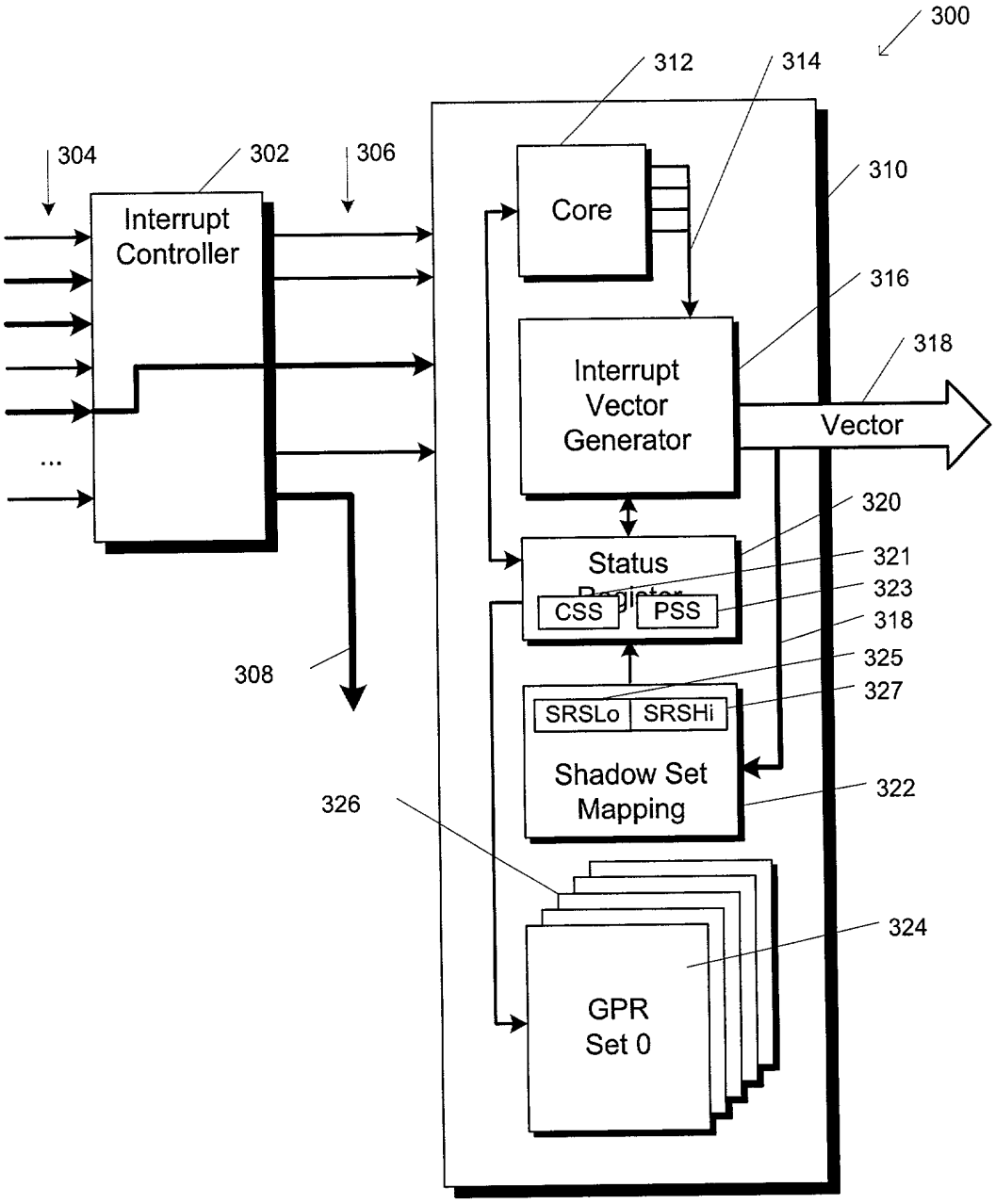

300
312
314
304
302
306
310
Core
Interrupt Controller
316
Interrupt Vector Generator
318
Vector
320
Status
321
CSS
PSS
323
318
325
SRSLo
SRSHi
327
Shadow Set Mapping
322
326
324
GPR Set 0
308

Figure 4

400

General Purpose Registers

Register Number	Name	Used For
0	zero	Always returns 0
1	at	(assembler temporary) Reserved for use by assembler
2-3	v0,v1	Value returned by subroutine
4-7	a0-a3	(arguments) First few parameters in a subroutine
8-15	t0-t7	(temporaries) Subroutines can use without saving
24,25	t8,t9	
16-23	a0-a7	Subroutine register variables; a subroutine that writes one of these must save the old value and restore it before it exits, so the calling routine sees the values preserved
26,27	k0,k1	Reserved for use by interrupt/trap handler
28	gp	Global pointer
29	sp	Stack pointer
30	s8/fp	Ninth register variable; subroutines that need one can use this as a frame pointer
31	ra	Return address for subroutine

Figure 5

500

Status Register Format - Status1

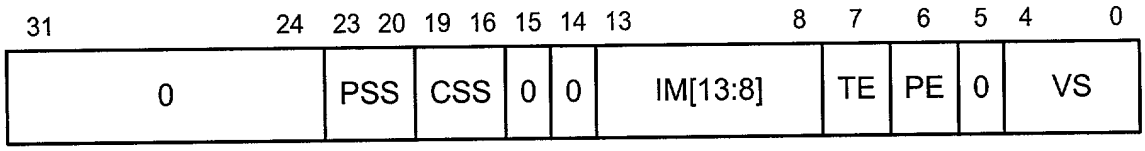


Figure 7

700

SRSLo Register Format

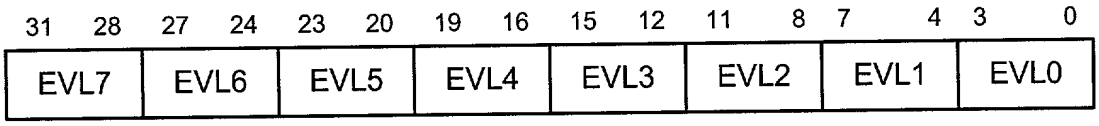


Figure 9

900

SRSHi Register Format

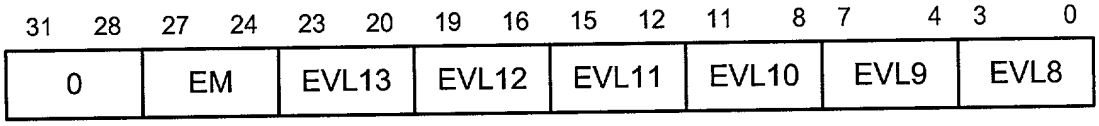


Figure 6a

600

Status1 Register Field Descriptions

Fields		Description	Read/ Write
Name	Bits		
0	31..204 15..14, 5	Must be written as zero; returns zero on read	0
PSS	23..20	Previous Shadow Set. If GPR shadow registers are implemented, this field is copied from the CSS field when an exception or interrupt occurs and provides the value of CSS. An eret instruction copies this value back into the CSS field. If no GPR shadow registers are implemented, this field is ignored on writes and returns zero on read.	R/W
CSS	19..16	Current Shadow Set. If GPR shadow registers are implemented, this field is the number of the current GPR set. If no GPR shadow registers are implemented, this field is ignored on write and returns zero on read.	R/W
IM[13:8]	13..8	Interrupt Mask: Controls the enabling of the IP[13:8] interrupts. **Encoding** / **Meaning** 0 — Interrupt request disabled 1 — Interrupt request enabled	R/W
TE	7	Timer Exclusive. This bit determines whether the timer interrupt is combined in an implementation dependent way with hardware interrupt 5 to become IP5 (default) or redirected to IP12. **Encoding** / **Meaning** 0 — Timer interrupt combined as IP5 1 — Timer interrupt redirected to IP12	R/W

Figure 6b

600

Status1 Register Field Descriptions, cont.

<table>
<tr><th colspan="2">Fields</th><th rowspan="2">Description</th><th rowspan="2">Read/ Write</th></tr>
<tr><th>Name</th><th>Bits</th></tr>
<tr><td>PE</td><td>6</td><td>Performance Counter Exclusive. This bit determines whether the performance counter interrupt is combined in an implementation dependent way with hardware interrupt 5 to become IP5 (default) or redirected to IP13.
<table><tr><th>Encoding</th><th>Meaning</th></tr><tr><td>0</td><td>Timer interrupt combined as IP5</td></tr><tr><td>1</td><td>Timer interrupt redirected to IP13</td></tr></table></td><td>R/W</td></tr>
<tr><td>VS</td><td>4..0</td><td>Vector Spacing. This field specifies the spacing between vectored interrupts and exceptions.
<table><tr><th>Encoding</th><th>Spacing Between Vectors (decimal)</th></tr><tr><td>16#00</td><td>0</td></tr><tr><td>16#01</td><td>32</td></tr><tr><td>16#02</td><td>64</td></tr><tr><td>16#04</td><td>128</td></tr><tr><td>16#08</td><td>256</td></tr><tr><td>16#10</td><td>512</td></tr><tr><td colspan="2">All other values are reserved.</td></tr></table></td><td>R/W</td></tr>
</table>

Figure 8

800

SRSLo Register Field Descriptions

Fields		Description	Read/ Write
Name	Bits		
EVL7	31..28	Shadow register set number for Exception Vector Level 7	R/W
EVL6	27..24	Shadow register set number for Exception Vector Level 6	R/W
EVL5	23..20	Shadow register set number for Exception Vector Level 5	R/W
EVL4	19..16	Shadow register set number for Exception Vector Level 4	R/W
EVL3	15..12	Shadow register set number for Exception Vector Level 3	R/W
EVL2	11..8	Shadow register set number for Exception Vector Level 2	R/W
EVL1	7..4	Shadow register set number for Exception Vector Level 1	R/W
EVL0	3..0	Shadow register set number for Exception Vector Level 0	R/W

Figure 10

1000

SRSHi Register Field Descriptions

Fields		Description	Read/ Write
Name	Bits		
0	31..28	Must be written as zero; returns zero when read	R/W
EM	27..24	Shadow register set number for non-vectored Exception Mode ($Status_{EXL}$=1). This value is used only if an interrupt is not serviced thru the vectored exception table.	R/W
EVL13	23..20	Shadow register set number for Exception Vector Level 13	R/W
EVL12	19..16	Shadow register set number for Exception Vector Level 12	R/W
EVL11	15..12	Shadow register set number for Exception Vector Level 11	R/W
EVL10	11..8	Shadow register set number for Exception Vector Level 10	R/W
EVL9	7..4	Shadow register set number for Exception Vector Level 9	R/W
EVL8	3..0	Shadow register set number for Exception Vector Level 8	R/W

Figure 11

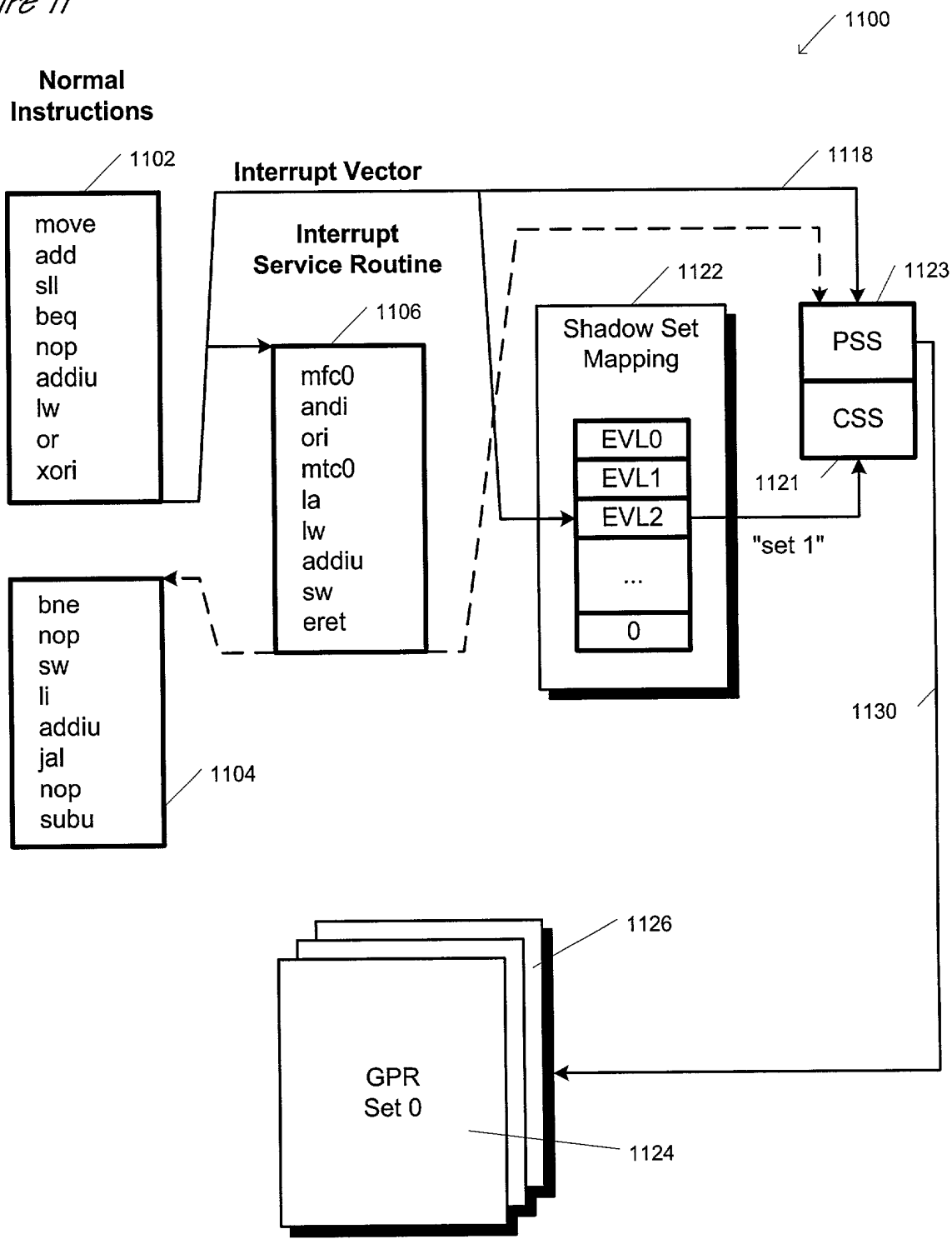

1100
Normal Instructions
1102
move
add
sll
beq
nop
addiu
lw
or
xori
Interrupt Vector
1118
Interrupt Service Routine
1106
mfc0
andi
ori
mtc0
la
lw
addiu
sw
eret
1122
Shadow Set Mapping
EVL0
EVL1
EVL2
...
0
1123
PSS
CSS
1121
"set 1"
bne
nop
sw
li
addiu
jal
nop
subu
1104
1130
1126
GPR Set 0
1124

Figure 12

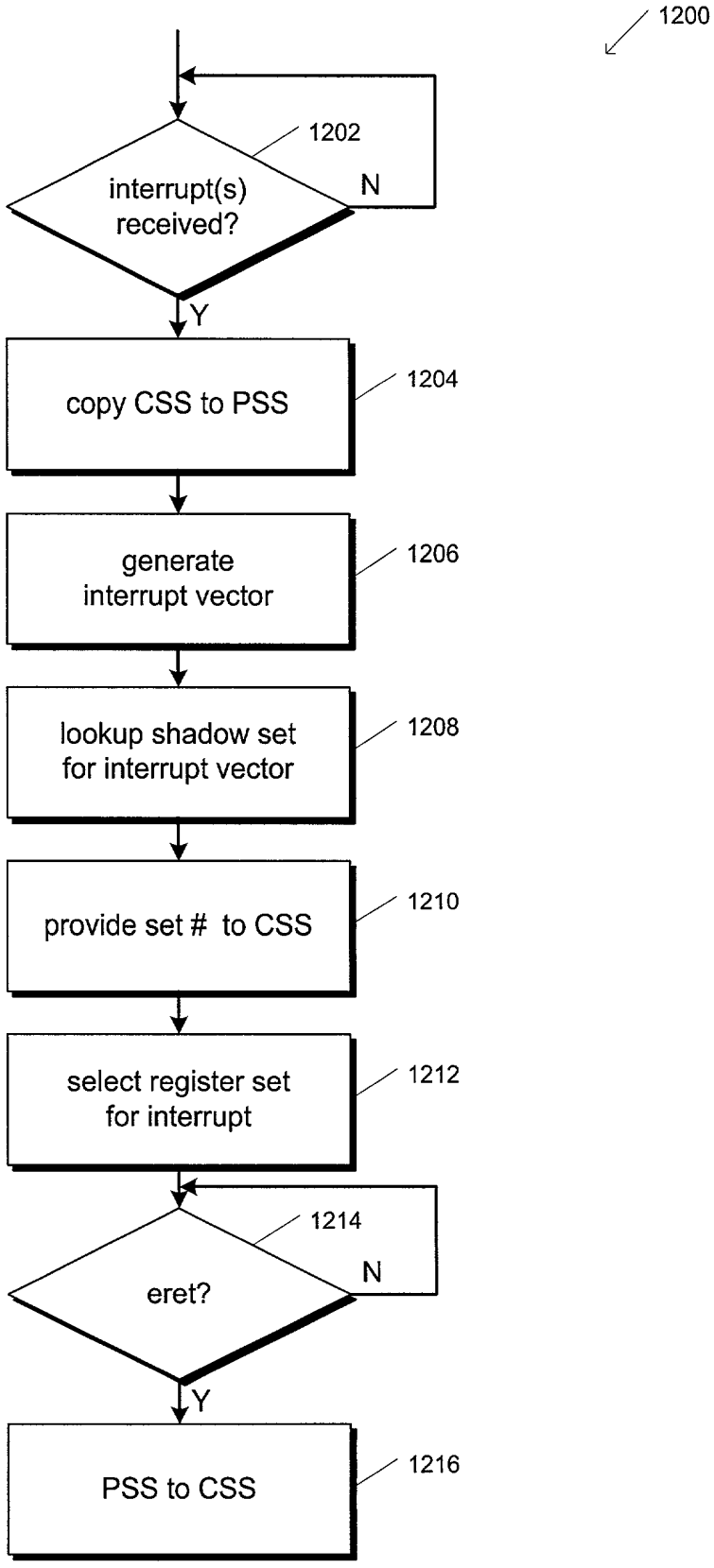
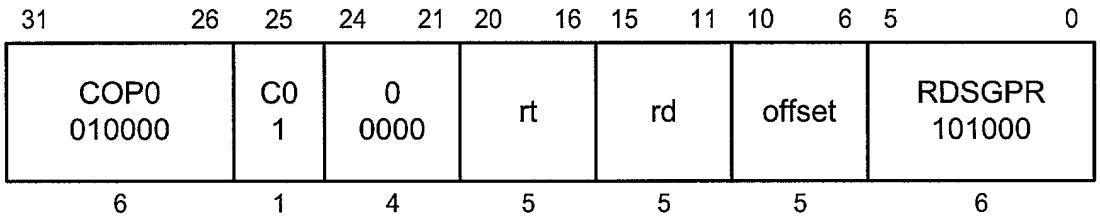

1200
1202
interrupt(s) received?
N
Y
copy CSS to PSS
1204
generate interrupt vector
1206
lookup shadow set for interrupt vector
1208
provide set # to CSS
1210
select register set for interrupt
1212
1214
eret?
N
Y
PSS to CSS
1216

Figure 13

1300

Read Shadow GPR Instruction - RDSGPR



Format: RDSGPR rt, offset(rd)

Purpose:
To move the contents of a shadow GPR register to a current GPR.

Description: rt ← SGPR [rd + offset]
The contents of the shadow GPR register specified by the sum of offset and rd is moved to the current GPR rt.

Restrictions:
The results are UNDEFINED if the sum of offset and rd do not reference an implemented shadow register.

Operation:

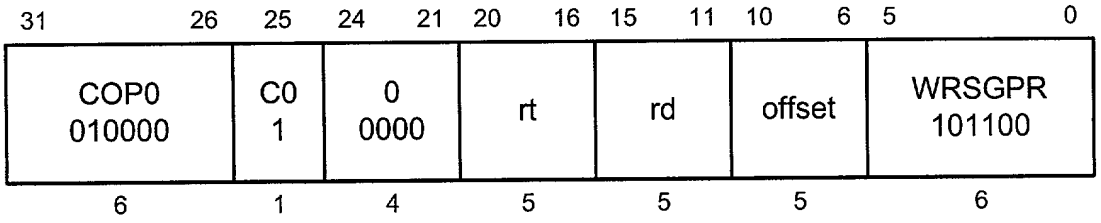
```
if IsCoprocessorEnabled(0) then
        if (ArchitectureRevision>=2) then
                GPR[rt] ← SGPR[rd = offset]
        else
                SignalException(ReservedInstruction)
        endif
else
        SignalException(CoprocessorUnusable, 0)
endif
```

Exceptions:
Coprocessor Unusable
Reserved Instruction

Figure 14

1400

Write Shadow GPR Instruction - WRSGPR



Format: WRSGPR rt, offset(rd)

Purpose:
To move the contents of a current GPR to a shadow GPR.

Description: SGPR [rd + offset]←rt
The contents of the current GPR rt is moved to the shadow GPR register specified by the sum of offset and rd.

Restrictions:
The results are UNDEFINED if the sum of offset and rd do not reference an implemented shadow register.

Operation:

```
if IsCoprocessorEnabled(0) then
        if (ArchitectureRevision>=2) then
                SGPR[rd = offset] ← GPR[rt]
        else
                SignalException(ReservedInstruction)
        endif
else
        SignalException(CoprocessorUnusable, 0)
endif
```

Exceptions:
Coprocessor Unusable
Reserved Instruction